Active Learning in Small to Large Courses

Owen L. Astrachan, Robert C. Duvall, Jeff Forbes, Susan H. Rodger

Abstract *—This paper presents our experiences promoting active learning in programming courses from introductory to advanced levels. We use a variety of techniques as our courses vary greatly in size and our facilities vary in layout and equipment. For large lectures, we present active interludes that require students to work in small groups, respond to periodic polls, or help a professor program. For moderately sized courses, we ask students to work in groups and share their observations with the class. Finally, in our Interactive Computer Classroom we have almost completely departed from long lectures to run the course in a workshop format, giving students a chance to work on the computer almost everyday in a supervised, safe environment. In short, although these techniques often require longer preparation time, we show that active learning can be done in any classroom situation and students must be active everyday to remain engaged in the material.*

Keywords —Active learning, computer science education, cooperative learning.

INTRODUCTION

Students have several obstacles to overcome in their introductory programming courses: learning about computer science, learning about program design (structured and object-oriented), learning about programming, and, in some cases, learning to be a student at a college/university. To help this process, we have developed a variety of techniques that encourage students to collaborate in problem solving during their scheduled lecture time. Research in cooperative and active learning literature [1], [2], [3], [4], [5], shows that students master material to a greater degree and retain more information when *active learning* is incorporated into the classroom. Students engaged in active learning may write, discuss, or attempt problem-solving during class as an alternative or in addition to lecture.

We tell students they cannot learn to program simply by reading and studying the concepts; they must practice programming as one would practice playing a sport or musical instrument. However, traditionally, only separate closed labs [6], [7] have been used to give students supervised, hands-on experience with conceptual material, and the typical closed lab does not involve any dynamic exchange among students or between teacher and student. In contrast, interactive lectures are common in other disciplines such as humanities. For example, in a foreign language course, students and the instructor engage in conversation most of the class period, either discussing exercises or acting out real-life situations. In fine arts, studio sessions are held in which students review their peers.

The next section describes the structure of our courses and the facilities which support them. Then two methods of active learning are discussed: one that closely mirrors traditional lecture styles and one that reduces lecturing to a minor component. Finally, several issues specific to large courses are discussed including ways to encourage equal student participation. In each case several alternatives are presented, each tailored to achieve our goals given different course environments.

STRUCTURE OF OUR COURSES

At Duke University the computer science undergraduate courses range in size from 15 to 250 students. Currently, the introductory sequence of Computer Science courses for majors, CompSci 6 and CompSci 100 (ACM CS 1 and 2 [8]), are taken by more than 500 students per year. Our non-majors courses, CompSci 1, CompSci 4 and CompSci 49S are taken by nearly 400 students each year, with the majority of students taking CompSci 1. CompSci 1 (ACM CS 0) is an overview of computer science with a small programming component, CompSci 4 is a non-majors programming course, and CompSci 49S is a special topics course for freshmen limited in size to 15. Our upper-level undergraduate courses usually have between 30 and 80 students. In total more than 1,500 students take a Computer Science course at Duke University each year.

Large introductory courses such as CompSci 1 and CompSci 6 are structured to have a lecture two to three days per week and a small lab once a week. To alleviate the burden of such large courses, we have sometimes been able to offer multiple sections of the same course that have different lecture times, but share all other out-of-class work. Some smaller courses with enrollments less than 40 (and a few sections of CompSci 6) are taught in our Interactive Computer Classrooms, ICCs, with computers for students to use during class. Many of our courses with size 50 or more either have the small lab once a week or smaller recitation sections once a week. All computer science courses at Duke University are taught by faculty. Graduate teaching assistants, TAs, usually lead the recitations and undergraduate teaching assistants, UTAs, lead the labs.

Structure of our Classrooms

Three different types of facilities are used for Computer Science courses. An auditorium is the largest size room with maximum seating from 100 to 300 seats. The design of an auditorium is usually many long rows of fixed seats that are

Duke University, Durham, NC 27705, csed@cs.duke.edu

progressively more elevated from the front to the rear. A regular classroom has maximum seating size of 20 to 80 seats and usually has a grid of fixed or movable seats that are not elevated. These classrooms may have long tables or desks on which the students can work. Our third type of room, an ICC, is a room with many computers designed for students to use the computer during class. All classrooms have Internet connections and one computer in each classroom can be projected on a large screen.

There are three ICCs at Duke University, one large and two small. The large ICC has 20 computers and 40 seats for students, while the smaller ICCs are have only 8 computers with 16 seats. Roughly half of the ICC is designed for using the computers (for the large classroom, either 20 students working alone or 40 students working in pairs) and half the room is designed for working in small groups at movable tables. Figure 1 shows a layout of the large ICC. There are 20 rectangular tables, each with one computer (shown as a square) and two seats (shown as circles). In addition there is a printer table (labeled P), a teacher's console (labeled TC) and 4 movable work tables. The work tables can be spread apart for group work.

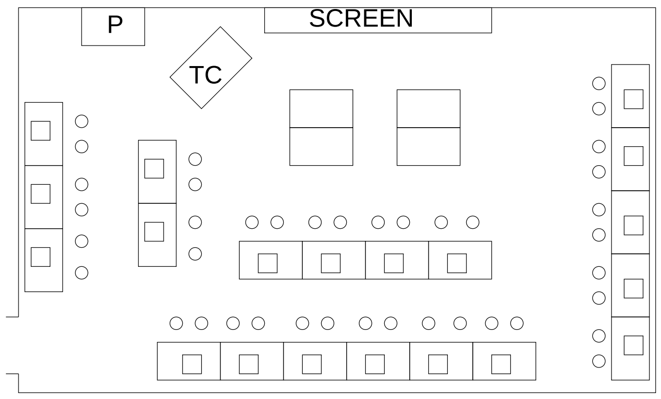

FIGURE 1
LARGE ICC LAYOUT

The ICCs are designed to help instructors control the room by arranging the computers in a U-shape around the sides and back of the room. From the front of the room, the instructor can view all monitor screens and can see quickly if anyone is falling behind or having difficulty. The students face the front of the room for discussion and rotate (facing the back or sides of the room) to use the computer. Since the students must turn 90 or 180 degrees to face the front for a discussion, the instructor can easily tell if anyone is not paying attention.

Several computer science courses have been taught in an ICC, CompSci 4 and CompSci 49S, and sections of CompSci 6 and CompSci 100. Other disciplines also teach in the large ICC including mathematics, statistics, biology, sociology, religion, economics, film and video, cultural anthropology, and French. The writing program for first-year students is the department primarily using one of the small ICCs.

ACTIVE LEARNING TECHNIQUES IN CLASS

Two styles of using standard lecture time designed to achieve more student involvement are presented: active lectures giving students a chance to check their understanding during the lecture by working on short exercises; whereas workshops allow students to work in a supervised, safe lab setting where the instructor lectures only briefly. However, the styles presented below can be used in traditional or ICCs.

Active Lectures

The term *active lecture* in this paper refers to a simple use of active learning similar to the lecture style used by most university educators modified to include other recent attempts to bring active learning to science classes [9], [10], [11].

An active lecture begins with a story that relates the day's topics to a real-world/industrial experience that will be familiar to students. For example, in discussing compression and Huffman coding in CompSci 100, the introduction discusses Napster, MP3, and lossy/lossless JPEG/GIF compression. These discussions focus not only on technical content, but on ethical and pragmatic concerns in using Napster. After the opening story, material is presented in a traditional lecture format.

The key element of an active lecture is a *reflective pause* or interlude. Stopping a lecture after 15 minutes, pausing a minute, and then proceeding increases student comprehension [12]. A reflective pause is directly related to the part of the lecture just given, but engages students in answering a question or solving a problem related to the lecture. For example, in the lecture on Huffman coding students are shown how an optimal tree/trie is constructed for encoding characters. The lecture discusses the greedy algorithm used for the construction, properties of greedy algorithms in general, and the specific code used in generating a tree from a sequence of characters. During the reflective pause students are asked two questions to test mastery and comprehension: uncompress a sequence of 30 bits given a diagram of a tree storing 8 characters; and think about creating change using a minimal number of coins when there are no nickels. The first problem tests basic understanding of how information is stored in a Huffman tree. The second requires students to think about possible alternatives when a greedy algorithm does not work in certain situations. Both questions can be answered by students individually in one or two minutes per question. Some professors have students discuss the answers, but this is not an integral part of an active lecture or a reflective pause.

A typical lecture is 50-75 minutes. This leads to dividing a lecture topic into two or three 15-20 minute chunks with each chunk followed by a reflective pause. Each pause requires a follow-up to discuss the answers and to ensure students have a basic understanding of materials before proceeding.

In the beginning programming courses, simply coding in front of the students can be very beneficial and leads to quality discussions. It helps students better understand the programming process because they see typos, compiler errors, and other problems encountered when programming; it makes them more attentive because they try to find our mistakes before the compiler does and it seems to make them more experimental and willing to ask "what happens when if". Using this technique, we have created a didactic form of pair programming in our large lecture courses. The instructor is the driver while the class as a whole (from 40 to 180 students) works together as the navigator of the pair-programming team [13], [14].

Workshops, not just Labs

Another type of active learning uses the workshop format and can be done in large or small courses during lecture or in an ICC. Each workshop period consists of a sequence of modules. Each module contains a mini-lecture (5-15 minutes) followed by an exercise to be performed by the students in groups, followed by a wrap-up discussing the different approaches students explored. The format also allows for *just-in-time teaching* [10] where the lecture is changed to address concerns that arise as a result of each workshop experience.

While students are working in class, it is important to give them as much quality feedback as possible. The most effective way we have found to do this is to move throughout the classroom visiting with each group or even just observing their discussions. However, as the class size increases, it becomes less practical for one person to reach every group. To alleviate this problem, UTAs are used during class to help visit groups and answer questions. Many UTAs are not prepared to support students well in these situations [15], so UTAs attend weekly meetings to discuss teaching techniques and to outline important issues to be covered in future classes. Additionally, UTAs provide exercises and bring up issues that are meaningful from their experience taking the class previously.

For example, for CompSci 6 in the ICC, we use the workshop format with one module per meeting. Class begins with a short review of the last module and an overview of the new activities for that day. The students work in pairs (one computer per pair) on the discovery exercises for most of the class period. The instructor and teaching assistants actively monitor the room helping out and offering advice. If many students are having the same problem, all students are asked to stop working and a short discussion is held. At the end of class, a wrapup discussion is always held.

Here is an example of a CompSci 6 lecture using C++. In the third lecture, students are given a 10-15 minute lecture on parameters, function signatures and input. They are then given the following program.

```
void Sweeps (string city)
{
   cout << "Amanda Harris, you've won TEN "
        << "million dollars!"
        << endl;
   cout << "Amanda, that's what you will hear "
        << "if you have the winning ticket."
        << endl;
   cout << "Imagine, Amanda, you could be "
        << "cruising in a Porsche down Main "
        << "Street in " << city << "."
        << endl;
}

int main ()
{
     Sweeps("Georgetown");
     return 0;
}
```

There are several modifications students are asked to make. The first one adds two more string parameters for first name and last name to the `Sweeps` function. Next they call the `Sweeps` function twice, once with each of their names and home towns. Next they do several exercises intended to cause errors, change one of the calls to pass just one parameter, and pass the parameters in a different order. Then they add a string variable to `main` for the first name and input the name. Next they add statements to read in the last name and city. That ends the first exercise. There are two more exercises of modifying programs and a fourth optional exercise for those students who complete the earlier exercises more quickly. Work is checked off by the instructor or UTAs either at the end of class or at the beginning of the next class to allow those students who do not finish during class to finish outside of class.

For classes not run in an ICC, many of these exercises are simulated using pencil and paper. Students are given a worksheet with code and modifications to be done and give them time to work on these problems during lecture. These exercises are different from typical closed lab sessions in a number of ways that help encourage a more collaborative atmosphere: often students are working in pairs to complete the exercises; the course staff is actively moving around the classroom eavesdropping and giving timely feedback; and often, student solutions are presented to the class for discussion.

We have found the last technique better than traditional techniques for engaging students attention by giving them investment in the material presented. In the past, even when group activities were used there was little feedback provided to students about their solutions nor was there any way for them to test their solutions. In general, there may be several solutions to a given problem, but the instructor only had time to present one or two solutions. Students were frustrated if their solution was different because they have no method for determining if their solution is correct. Now, more time is devoted to showing student solutions which usually vary more than the instructor's examples. And, the process of viewing multiple solutions invariably provides an easy way to discuss all of the pedagogical points we would have wanted to show.

Students show their work in a number of ways: write their solutions on an overhead transparency with the pens provided, write their solutions on ordinary notebook paper which is displayed using a document viewer, or simply call out the next line of code for the instructor to type.

MAKING LARGE LECTURES MORE ACTIVE

In teaching a large introductory computer science class, it is especially hard to make lectures more active and create a better learning experience [16]. Among the main problems with large lectures are: gauging how well the course material is being understood; creating a more engaging and enjoyable learning environment; and encouraging students to come to class on time and rewarding those that do come and participate.

Encouraging Participation

One step towards addressing these issues we have developed is the pick-a-student system. An important part of asking a question is to get everyones' attention and not have a dialogue with only the few students who always volunteer. At any point in lecture, if there is a question to be asked about the material or if a student's work is to be selected to be displayed on the screen, the pick-a-student system randomly selects a student and displays their picture and name on the screen. If the selected student answers the question, they receive credit and their picture and name are removed from the pool. The pick-a-student must be sufficiently employed through the semester to cycle through all of the students in the course at least once. This requirement should encourage the instructor to ask many questions and receive feedback from a wide cross-section of the students.

It is vitally important students know it is reasonable not to know the answer and that any attempt is worthwhile. When students have no idea whatsoever, they have several options: simply say "I don't know"; ask a question back to the instructor that may help elucidate the problem; or call upon another student in the class, or even their entire row, to help them.

There are a number of added benefits to the pick-a-student system. It makes it much easier for the instructor to learn students' names and that in turn seems to make students more comfortable interacting with the instructor both during and outside of class. Students also seem to enjoy the process of seeing their faces appear on the big screen and wondering if their name will be chosen. Presumably, it has a similar draw as games of chance. We have even used the pick-a-student system as a later assignment in our CompSci 1 course that allows students to experiment with randomness and arrays.

The pick-a-student system is fairly easy to create if you can get pictures of the students along with their names. For a yearly cost, the university can provide ID pictures for each student. Instead, we have tried a more playful approach by passing around a digital camera and sheet of paper on the first day of class. The instructor takes a picture of the first student, and each student subsequently takes a picture of another student, with the last student taking a picture of the instructor. Students sign the paper in the order their pictures are taken. Once the images have been saved with the students names, there are several ways to display them: we have written a simple program that reads in the directory of pictures and flips through the pictures, randomly stopping on a student; or on Windows and Macintosh computers, standard slide show software can be used to flip through a folder of images.

Group Activities

Organizing students into groups can lead to a more productive use of class time and can be beneficial to students in many ways. We assign pairs of students in small courses and assign groups of size 3 to 5 students in larger courses. Students receive a seating chart with their name and their assigned seat and are expected to sit there. Groups are changed about once a month, or about 4-6 times during the semester.

There are several advantages for instructors to assign students seating. For large courses, having assigned seats makes it possible to hand out homework during class. The homework is sorted by seating and can be passed out quickly. This task would take a very long time in a course of 200 students if each student's name was called out and they had to walk up to the front to pick up their work. Having assigned seats also makes it easier for the instructor to get to know the students because they sit in the same place and he/she can look up their name.

Assigned seating and groups and changing them often has benefits to students as well. Students will get to know more people in the course as they are forced to meet people they might not otherwise meet. Students are more likely to volunteer to answer a question if they feel they are part of a group and the group has had a chance to discuss the answer. The student is comfortable because the group agreed on the answer. Some groups have become very competitive, such as bringing in their group number on a fancy sign and holding it up high whenever they answered a question.

SUPPORTING ACTIVE LEARNING

There are several impediments to wide-spread adoption of active learning techniques by educators [17]: the time and preparation required to develop materials; the disparity between active learning and the educational experience of most academics some of which is tied to ceding some control in the classroom if lectures are abandoned; the perception that active lectures are too slow compared to standard lectures, making it harder to cover as much material; and the difficulty of ensuring students come to class prepared. Since we believe students must be active everyday to remain engaged in the material, we have developed a few techniques to help address these issues.

There is no doubt preparing for active learning takes more time from both instructors and students. Instructors must be better prepared because the material needed in class may be more varied as students bring up issues from the reading or problems they are asked to solve. Students, as well, must prepare before class by doing the assigned reading or preparatory work or they will be unable to participate in class. When students are unable to be actively involved in the material the class typically reverts back to a lecture format.

One step towards ensuring students are prepared is pre-class quizzes, available before each lecture that must be completed by the time students enter the class. These quizzes are meant to be fairly simple if the student has read the reading material and difficult for students to do correctly by simply guessing. This forces students to read or skim reading material, making them better prepared to answer questions during class. An additional feature is that the instructor can look over a summary of the answers to quizzes before heading to class and can see which topics students understand and which ones students need further explanation, allowing the instructor to adjust the focus of the lecture and, perhaps, skip or skim material that has been mastered

Currently, we use the Blackboard CourseInfo system [18] to administer online multiple choice quizzes. An advantage of this system is that one can designate a specific time range during which the quiz can be taken. Previously, paper quizzes were given that were due when the students walked in the door. A database of collected quiz questions is available from Ed Gehringer [19].

CONCLUSIONS

Collaborative learning in small groups increases interest and participation in class. We have observed that giving students time to think during class results in more student participation in discussions, and that discussions are more meaningful, especially in uncovering misconceptions and misunderstandings. This approach also increases student confidence and self-esteem. We have observed that students who were reluctant to speak out in class are now eager to volunteer and discuss a solution that their group developed. In evaluations of our classes using collaborative learning, students overwhelmingly stated that they preferred the group problem solving format to the lecture format. In addition, female students (most classes contain 25-30% women) made comments that rotating assigned groups gave them a chance to meet other students in the class that they would not have approached otherwise, making them feel more comfortable in the classroom. These findings reflect those of other researchers [20].

Although less material may be presented during class, we find that students understand the material more thoroughly. We make up for our decreased pace by placing more responsibility on students to work outside of lecture to prepare for class. We ensure that they are not missing the basics by keeping tabs on their progress and adjusting our presentations to match their progress and not going over material it is clear they have mastered.

We continue to make our courses more active. In the past we have integrated visual and interactive computer tools into the classroom [21] and toys into lecture [22]. More recently we have found more effective ways to use computers in the classroom to change the way in which we present material to the students. However, almost surprisingly, we have also found parallel activities that do not use computers directly but achieve similar advantages.

REFERENCES

[1] David Johnson, Rodger T. Johnson, and Karl A. Smith, *Cooperative learning: Increasing College Faculty Instructional Productivity,* Volum 20, Number 4, ED347871. ASHE/ERIC Higher Education, 1991.

[2] Robert E. Slavin, *Cooperative Learning: Theory, Research, and Practice,* Allyn and Bacon, second edition, 1995.

[3] Charles C. Bonwell and James A. Eison, *Active Learning: Creating Excitement in the Classroom,* ED336049. ASHE-ERIC Higher Education Reports, 1991, http://ericae.net/db/edo/ED34027.htm.

[4] J. G. Penner, *Why Many College Teachers Cannot Lecture,* Charles C. Thomas, 1984.

[5] Tom Drummond, "A brief summary of the best practices in college teaching," http://nsccux.sccd.ctc.edu/∼eceprog/bstprac.html, 1995.

[6] Deborah Knox, Ursula Wolz, Daniel Joyce, Elliot Koffman, and Joan Krone, "Use of laboratories in computer science education: Guidelines for good practice: Report of the working group on computing laboratories," *Proceedings of the Conference On Integrating Technology Into Computer Science Education,* pp. 167–181, 1996.

[7] Alan Fekete and Antony Greening, "Designing closed laboratories for a computer science course," *Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education,* pp. 295–299, 1996.

[8] ACM Curriculum Committee on Computer Science, "Curriculum '78: Recommendations for the undergraduate program in computer science," *Communications of the ACM,* vol. 22, no. 3, pp. 147–166, 1979.

[9] Eric Mazur, *Peer Instruction: A User's Manual,* Prentice Hall, 1996.

[10] Gregor M. Novak, Evelyn T. Patterson, Andrew D. Gavrin, and Wolfgang Christian, *Just-In-Time Teaching : Blending Active Learning With Web Technology,* Prentice Hall College Division, 1999.

[11] Pedagogical Patterns, ," http://www.pedagogicalpatterns.org/.

[12] Kathy L. Ruhl, Charles A. Hughes, and Patrick J. Schloss, "Using the pause procedure to enhance lecture recall," *Teacher Education and Special Education,* vol. 10, pp. 14–18, 1987.

[13] Owen L. Astrachan, Robert C. Duvall, and Eugene Wallingford, "Bringing extreme programming to the classroom," *2001 XP Universe Conference Proceedings,* 2001, http://www.xpuniverse.com/.

[14] Laurie Williams and Robert R. Kessler, "Experimenting with industry's pair-programming model in the computer science classroom," *Journal on Computer Science Education,* 2001.

[15] Dale Winter, Paula Lemons, Jack Bookman, and William Hoese, "Novice instructors and student-centered instruction: Identifying and addressing obstacles to learning in the college science laboratory," *Journal of Scholarship of Teaching and Learning,* vol. 2, no. 1, 2001, http://www.iusb.edu/∼josotl/.

[16] S. Wolfman, "Making lemonade: Exploring the bright side of large lecture classes," *Thirty-third SIGCSE Technical Symposium on Computer Science Education,* pp. 257–261, 2002.

[17] Joseph Lowman, *Mastering the Techiques of Teaching,* Jossey-Bass, 1984.

[18] Blackboard, ," http://www.blackboard.com/.

[19] Edward F. Gehringer and Tony M. Louca, "A web-based object technology course database," *2000 OOPSLA Conference Proceedings,* 2001, http://www.xpuniverse.com/.

[20] J. D. Chase and Edward G. Okie, "Combining cooperative learning and peer instruction in introductory computer science," *Thirty-First SIGCSE Technical Symposium on Computer Science Education,* pp. 372–376, 2000.

[21] Susan H. Rodger, "An interactive lecture approach to teaching computer science," *Twenty-Sixth SIGCSE Technical Symposium on Computer Science Education,* pp. 278–282, 1995.

[22] Owen L. Astrachan, "Concrete teaching: Hooks and props as instructional technology," *Proceedings of the Third Annual Conference On Integrating Technology Into Computer Science Education,* pp. 21–24, 1998.